UNIVERSIDADE FEEVALE

FELIPE FERNANDES LORENZONI

COMUNICAÇÃO ENTRE BACK-END E FRONT-END NO HEALTH SIMULATOR

(Título Provisório)

Anteprojeto de Trabalho de Conclusão

Novo Hamburgo

2016

FELIPE FERNANDES LORENZONI

COMUNICAÇÃO ENTRE BACK-END E FRONT-END NO HEALTH SIMULATOR

 (Título Provisório)

Anteprojeto de Trabalho de Conclusão de Curso, apresentado como requisito parcial

à obtenção do grau de Bacharel em

Sistemas de Informação pela

Universidade Feevale

Orientadora: Marta Rosecler Bez

Co-Orientador: Paulo Ricardo Muniz Barros

Novo Hamburgo

2016

RESUMO

O ensino na área da saúde vem se beneficiando de forma expressiva de aplicações que simulam casos clínicos. O *Health Simulator* é um simulador do tipo Paciente Virtual (PV) que ajuda a diminuir a lacuna que existe entre o aprendizado teórico e a grande quantidade de informações que o aluno deve absorver. O simulador é separado em duas partes: o *front-end,* que contempla o aplicativo do jogo e o *back-end*, que abrange o servidor que armazena as informações trocadas entre o jogo e o servidor. O objetivo principal deste trabalho é desenvolver a integração entre estas duas partes. O meio de comunicação que está sendo desenvolvido se utiliza de um Serviço *Web* que é comumente utilizado para soluções complexas de comunicações entre diferentes aplicações, que dispõem de uma solução que visa obter uma maior escalabilidade e que diferentes plataformas possam se beneficiar desta solução. Desse modo, optou-se pela arquitetura *REST* (*Representational State Transfer*) entre as aplicações cliente (jogo/cliente) e um fornecedor do serviço (*back-end*). Aplicações que utilizam uma arquitetura *REST* são denominadas de *RESTful*. Uma vez estipulado que o sistema que será desenvolvido seguirá uma arquitetura do tipo *REST*, este deve obedecer algumas restrições: implementar o tipo de comunicação cliente-servidor, ser *stateless*, utilização de recursos de *cache*, possuir uma interface uniforme, ser estruturado num sistema de camadas e, como uma restrição opcional, ter um plano de desenvolvimento de código sob-demanda. Desta forma, o trabalho proposto visa a implementar um modo de realizar a integração e comunicação entre as partes do simulador, buscando como um dos objetivos a escalabilidade do projeto, ou seja, estar disponível para o maior número possível de alunos.

Palavras chaves: *Health Simulator. REST. Web Service*. Escalabilidade. Simulação.

SUMÁRIO

MOTIVAÇÃO ...........................................................................................................................5

OBJETIVOS ..............................................................................................................................8

METODOLOGIA ......................................................................................................................9

CRONOGRAMA .................................................................................................................... 11

BIBLIOGRAFIA ....................................................................................................................13

MOTIVAÇÃO

O modo como o ensino na área da saúde é transmitido para os alunos nem sempre consegue atingir as “Diretrizes Curriculares Nacionais do Curso de Medicina”. Segundo Bez (2013), o currículo e o método pedagógico que se espera devem permitir aos alunos o desenvolvimento da capacidade de observar e de escutar. Com isso, tornar o estudante preparado para pensar e a aprender, ser, fazer e conviver com a sua aprendizagem.

Aspirando diminuir a lacuna que existe entre o aprendizado teórico e o grande volume de informações que este deve absorver, foi criado na universidade Feevale, um grupo de estudos intitulado de Computação Aplicada a Saúde (CAS), vinculado ao grupo de pesquisa de Computação Aplicada. Nesta equipe está sendo desenvolvido um projeto chamado *Health Simulator*.

Este projeto visa à implementação de um simulador do tipo paciente virtual (PV). Segundo Orton e Mulhausen (2008, p. 75), um Paciente Virtual (PV) é “*um programa interativo que simula a vida real em cenários clínicos, que permite o aprendizado de atos do profissional da saúde, obtendo a história clínica, exames e realizando diagnóstico e decisões terapêuticas*”.

O Health Simulator, devido a sua complexidade e diferentes arquiteturas, é divido em duas partes: o *front-end*, que contempla o aplicativo do jogo e o *back-end*, que abrange o servidor que armazena as informações trocadas entre o jogo e o servidor. Desta forma, é necessária uma estrutura de comunicação entre o *front-end* e o *back-end,* que será responsável pela troca de informações entre os dois ambientes. A plataforma de comunicação que está sendo desenvolvida faz uso de um Serviço *Web* (SOMMERVILLE, 2011) que possibilita uma solução com maior escalabilidade. Desse modo, optou-se pela arquitetura *REST* (*Representational State Transfer*) entre as aplicações cliente (jogo/cliente) e um fornecedor do serviço (*back-end*) (SOMMERVILLE, 2011). *REST* é um estilo arquitetural de sistemas hipermídia distribuídos (FIELDING, 2000). Aplicações que utilizam uma arquitetura *REST* são denominadas de *RESTful* (FIELDING, 2000).

Um *web service* *RESTful*, se comparado aos protocolos “WS-” (SOAP e WSDL são exemplos), tem a vantagem de aproveitar mais os recursos contidos no próprio protocolo HTTP (*Hypertext Transfer Protocol*), como códigos de retornos, cabeçalhos e verbos (WEBBER; PARASTATIDIS; ROBINSON, 2010). Dessa forma, simplifica a solução e pode diminuir a quantidade de dados trafegados, uma vez que existirá um nível a menos de abstração na implementação da comunicação.

Uma vez determinado que o sistema é *RESTful*, este deve obedecer a algumas restrições para alcançar uma alta escalabilidade no serviço (FIELDIN, 2000). Na arquitetura *REST* o modelo de troca de informações a ser utilizado é do tipo cliente-servidor (FIELDING, 2000). Neste modelo, o cliente iniciará uma ação de requisição de uma tarefa disponibilizada pelo servidor. O servidor, por sua vez, receberá a tarefa e devolverá para o cliente o seu resultado (TANENBAUM; STEEN, 2006).

Neste modelo proposto há uma distribuição clara da responsabilidade de cada lado. O servidor é responsável por prover e manter as informações aos clientes. Os clientes devem, de algum modo, realizar a conexão com o servidor para que possam utilizar a informação armazenada.

Como não se mantém uma relação entre os clientes, a tramitação entre soluções ocorre de maneira simplificada. Uma vez que não existe dependência entre os clientes, a sua portabilidade para diversas plataformas é facilitada. A incomplexidade do servidor agrega para a sua escalabilidade. Além disso, neste modelo é possível que os clientes e os servidores se desenvolvam de maneira distinta (FIELDING, 2000).

Em uma arquitetura *REST*, a comunicação entre cliente e o servidor deve ser *stateless*, isto é, nenhum estado da sessão pode ser mantido no servidor. Toda a chamada realizada para o servidor deve conter todas as informações para ser compreendida (FIELDING, 2000). Desse modo, obtém-se transparência, segurança e escalabilidade.

O *cache* é utilizado como recurso para que seja possível armazenar temporariamente informações, visando a um ganho de performance da rede (VELLOSO, 2014). As respostas enviadas pelo servidor devem ser sinalizadas de forma direta ou indiretamente como *cacheable* ou *non-cacheable*. No caso de uma resposta enviada como *cacheable*, existe um aumento da performance, uma vez que o cliente consegue reaproveitar as informações armazenadas para interações equivalentes (FIELDING, 2000). Deste modo, é possível eliminar interações, melhorando a eficácia, a escalabilidade e o desempenho notado pelo usuário.

Para tornar mais simples e melhorar a visibilidade das interações, um sistema *RESTful* necessita que todos seus componentes possuam uma interface uniforme (FIELDING, 2000). Além disso, a implementação não fica dependente de outros serviços disponibilizados, o que possibilita que ambas as partes consigam ser desenvolvidas de forma distinta. Entretanto, para a implementação de uma interface uniforme num sistema, este deve obedecer quatro restrições: identificação de recursos, manipulação de recursos por meio de representações, mensagens auto-descritivas e hipermídia como o motor do estado da aplicação (FIELDING, 2000).

O protocolo HTTP, teoricamente, é um entre outros protocolos que possibilita a implementação de uma interface uniforme, sendo assim, a maior parte das aplicações *RESTful* são construídas utilizando-o (WEBBER; PARASTATIDIS; ROBINSON, 2010). O reconhecimento de recursos, no protocolo HTPP, é realizado mediante URIs (*Uniform Resource Identifiers*) (STEPHEN, 2001).

Um recurso detectado por uma URI pode dispor de variados tipos de representação. O tipo de representação de um recurso não exige ser o mesmo no qual ele é armazenado no servidor. Este encapsulamento garante um baixo acoplamento (WEBBER; PARASTATIDIS; ROBINSON, 2010), uma vez que o tipo de armazenamento do recurso pode ser convertido para outro mais eficaz sem causar alterações no modo como os clientes manipulam o serviço.

Um sistema de camadas equivale a uma “torre de camadas”, onde cada uma é sobreposta à outra. A camada superior dispõe dos serviços da inferior a si, contudo, a inferior não tem percepção da superior. Uma camada só é exibida para as que consomem os serviços disponíveis por ela, isto é, em um grupo de camadas no qual a camada 3 usa serviços da 2, e a 2 usa serviços da 1, a 3 desconhece a existência da 1 (FOWLER et al, 2012). Deste modo, numa arquitetura *REST*, uma condição que deve ser atendida é que ela deve ser uma arquitetura de camadas e que cada componente não consegue acessar além da camada com a qual interage diretamente (FIELDING, 2000).

A última restrição numa arquitetura *REST* é a oportunidade de o cliente baixar um código (na forma de *applets* ou *scripts*) do servidor (FIELDING, 2000). Isto tem como objetivo simplificar a codificação do cliente, reduzindo o número de funções a serem desenvolvidas, além de melhorar a extensibilidade. Entretanto, é uma restrição opcional, haja vista diminuir a visibilidade do sistema. Sendo que um dos principais objetivos de um *web service,* que é utilizado numa parte significativa do processamento de informação do *Health Simulator*, é prover uma alta escalabilidade e estar acessível para o maior número de clientes possível, com estes atributos apresentados nesta seção, uma arquitetura *RESTful* se manifesta apropriada para o problema. Uma vez estabelecida a arquitetura a ser utilizada, será verificado se esta, se aplica da melhor forma possível para realizar a comunicação entre o *front-end e* o *back-end*, estes que são construídos em linguagens de programação distintas.

Portanto, este trabalho tem como questão de pesquisa: Como desenvolver uma camada de comunicação multiplataforma entre o *back-end* e *front-end*, do projeto *Health Simulator*, de forma a atender as questões relativas a interação de forma segura e escalável?

OBJETIVOS

**Objetivo geral**

Este trabalho tem como objetivo geral realizar a conexão entre o *front-end* e o *back-end* do projeto Health Simulator, utilizando princípios *REST* para realizar as trocas de dados e informações durante a utilização do simulador, de forma segura e eficaz.

**Objetivos específicos**

* Realizar um estudo bibliográfico sobre os temas pertinentes a este trabalho: *Health Simulator*, princípios de *Restful*, questões relativas a segurança na troca de informações, entre outros.
* Integrar o *front-end* com o *back-end* do projeto *Health Simulator*.
* Explorar e corrigir falhas de segurança que possam existir entre a comunicação do *front-end* com o *back-end*.
* Validar a comunicação entre o *back-end* e o *front-end*.

METODOLOGIA

A figura abaixo servirá para expor a metodologia que será aplicada neste trabalho, tendo os atributos que a definem destacadas em vermelho.



**Figura: Classificação da Pesquisa (adaptado de BEZ, 2011)**

Este trabalho se caracteriza como pesquisa aplicada, visto que seu fruto (i.e. o desenvolvimento de uma camada de integração entre o *back-end* e o *front-end* no *Health Simulator*) visa a uma solução de um problema de ordem prática (i.e. integração e comunicação entre as partes do simulador). Nele, serão abordados conhecimentos básicos, já concretizados na área de Sistemas de Informação, sem o objetivo de formular novos conceitos ou mudar algum paradigma já existente.

O projeto será abordado de forma qualitativa, pois a validação do trabalho será feita com base na conformidade da solução com o problema proposto, ou seja, a integração e comunicação no simulador. Para melhor compreensão das etapas do projeto, estas são identificadas na sequência:

* ETAPA 1: Estudo bibliográfico sobre os temas pertinentes a este trabalho: *Health Simulator*, princípios de *Restful*, questões relativas a segurança na troca de informações, entre outros.
* ETAPA 2: desenvolvimento do módulo de integração entre o *front-end* com o *back-end* do projeto Health Simulator.
* ETAPA 3: análise e correção de falhas de segurança que possam existir entre a comunicação do *front-end* com o *back-end*.
* ETAPA 4: validação da comunicação entre o *back-end* e o *front-end*.

CRONOGRAMA

Trabalho de Conclusão I

|  |  |
| --- | --- |
| Etapa  | Meses |
| Ago | Set | Out | Nov |
| Revisão bibliográfica sobre o *Health Simulator*. |  |  |  |  |
| Revisão bibliográfica sobre princípios REST |  |  |  |  |
| Revisão bibliográfica sobre questões relativas a segurança na troca de informações. |  |  |  |  |
| Escrita do anteprojeto. |  |  |  |  |
| Revisão do anteprojeto |  |  |  |  |
| Entrega do anteprojeto. |  |  |  |  |
| Escrita do TCI |  |  |  |  |
| Revisão do TCI |  |  |  |  |
| Entrega do TCI |  |  |  |  |

Trabalho de Conclusão II

|  |  |
| --- | --- |
| Etapa  | Meses |
| Mar | Abr | Mai | Jun |
| Desenvolvimento do módulo de integração entre o *front-end* com o *back-end* do projeto Health Simulator. |  |  |  |  |
| Análise e correção de falhas de segurança que possam existir entre a comunicação do *front-end* com o *back-end*. |  |  |  |  |
| Validação da comunicação entre o *back-end* e o *front-end*. |  |  |  |  |
| Escrita do TCII. |  |  |  |  |
| Revisão do TCII. |  |  |  |  |
| Entrega do TCC II. |  |  |  |  |
| Apresentação a banca avaliadora |  |  |  |  |

BIBLIOGRAFIA

FIELDING, R. T. ***Architectural Styles and the Design of Networkbased Software Architectures***. Doctoral dissertation, University of California, Irvine, 2000.

ORTON, E.; MULHAUSEN, P. *E-learning virtual patients for geratric education*. **Gerontology & Geriatrics Education**, 2008. v. 28, n. 3, p.73-88.

RICHARDSON, L.; RUBY, S. ***RESTful Web Services***. O’Reilly Media, 2007.

SOMMERVILLE, I. **Arquitetura orientada a serviços. Engenharia de Software**. 9th ed., p.355–368. São Paulo: Person Prentice Hall, 2011.

TANENBAUM, A. S.; STEEN, M. Van. ***Distributed Systems: Principles and Paradigms***. 2 ed. Pearson Prentice Hall, 2006.

VELLOSO, F. **Informática: Conceitos Básicos.** 9. ed. Rio de Janeiro, Elsevier, 2014.

WEBBER, J.; PARASTATIDIS, S.; ROBINSON; I. ***REST in Practice: Hypermedia and Systems Architecture***. O’Reilly Media, 2010.